The embodiments of the invention in which an exclusive property or privilege is claimed are defined as follows:

1. A computer-implementable method for determining the behavior of an executable comprising:

(a) selecting evaluation calls made by the executable to the interface of an operating system;

(b) loading stubs into a virtual address space, the stubs:

(i) mirroring the calls made to the interface of an operating system; and

(ii) determining a behavior signature for the selected calls;

(c) executing the selected calls inside of a virtual operating environment using the loaded stubs dynamically linked libraries; and

(d) determining the behavior signatures resulting from said execution of the selected calls inside of a virtual operating environment.

2. The method of Claim 1 wherein the calls selected for evaluation are a subset of calls made by the executable to the interface of an operating system.

3. The method of Claim 1 wherein the calls are application programming interface (API) calls.

4. The method of Claim 1 wherein the calls are included in dynamic link libraries (DLLs) and wherein loading stubs include loading stub DLLs into said virtual address space.

5. The method of Claim 4 wherein mirroring the calls made to the interface of an operating system includes mirroring a set of full implemented DLLs.

6. The method as recited in Claim 1 further comprising writing the behavior signature of the selected calls to an output store.

7. The method as recited in Claim 6 wherein writing the behavior signature to an output media includes writing three parameters to the output media comprising:

(a) a first parameter indicative of the call made to the operating system;

(b) a second parameter operative to store a variable of known data types; and

(c) a third parameter operative to store a variable of known data types.

8. The method as recited in Claim 2, wherein selecting a subset of calls includes:

(a) traversing the executable's machine code and identifying calls that are directed to the interface of the operating system; and

(b) identifying calls that are potentially indicative of malware.

9. The method as recited in Claim 8, wherein identifying calls that are potentially indicative of malware includes:

comparing calls made in the executable with calls that exist in known malware; and

if a call matches one that exists in known malware, determining that the call is potentially indicative of malware.

10. The method as recited in Claim 8, wherein identifying calls that are potentially indicative of malware includes:

comparing calls made in the executable with calls that are identified as a future security threat from malware; and

if a call matches one that is identified as a future security threat from malware, determining that the call is potentially indicative of malware.

11. The method as recited in Claim 2, wherein selecting a subset of calls includes determining if each call requires execution by a stub.

12. The method as recited in Claim 1, wherein loading stubs is initiated by an event generated by the virtual operating environment.

13. The method as recited in Claim 1, wherein loading stubs is performed by a loader that copies the stubs from a storage media to a virtual address space.

14. The method as recited in Claim 1, wherein loading stubs includes determining the stubs that handle the selected calls.

15. A computer-readable medium bearing computer-executable instructions which, when executed, carry out a method for determining the behavior of an executable comprising:

(a) selecting evaluation calls made by the executable to the interface of an operating system;

(b) loading stubs into a virtual address space, the stubs:

(i) mirroring the calls made to the interface of an operating system; and

(ii) determining a behavior signature for the selected calls;

(c) executing the selected calls inside of a virtual operating environment using the loaded stubs dynamically linked libraries; and

(d) determining the behavior signatures resulting from said execution of the selected calls inside of a virtual operating environment.

16. The computer-readable medium of Claim 15 wherein the calls selected for evaluation are a subset of calls made by the executable to the interface of an operating system.

17. The computer-readable medium of Claim 15 wherein the calls are application programming interface (API) calls.

18. The computer-readable medium of Claim 15 wherein the calls are included in dynamic link libraries (DLLs) and wherein loading stubs include loading stub DLLs into said virtual address space.

19. The computer-readable medium of Claim 18 wherein mirroring the calls made to the interface of an operating system includes mirroring a set of full implemented DLLs.

20. The computer-readable medium of Claim 15 further comprising writing the behavior signature of the selected calls to an output store.

21. The computer-readable medium of Claim 20 wherein writing the behavior signature to an output media includes writing three parameters to the output media comprising:

(a) a first parameter indicative of the call made to the operating system;

(b) a second parameter operative to store a variable of known data types; and

(c) a third parameter operative to store a variable of known data types.

22. The computer-readable medium of Claim 16, wherein selecting a subset of calls includes:

(a) traversing the executable's machine code and identifying calls that are directed to the interface of the operating system; and

(b) identifying calls that are potentially indicative of malware.

23. The computer-readable medium of Claim 22, wherein identifying calls that are potentially indicative of malware includes:

comparing calls made in the executable with calls that exist in known malware; and

if a call matches one that exists in known malware, determining that the call is potentially indicative of malware.

24. The computer-readable medium of Claim 22, wherein identifying calls that are potentially indicative of malware includes:

comparing calls made in the executable with calls that are identified as a future security threat from malware; and

if a call matches one that is identified as a future security threat from malware, determining that the call is potentially indicative of malware.

25. The computer-readable medium of Claim 16, wherein selecting a subset of calls includes determining if each call requires execution by a stub.

26. The computer-readable medium of Claim 15, wherein loading stubs is initiated by an event generated by the virtual operating environment.

27. The computer-readable medium of Claim 15, wherein loading stubs is performed by a loader that copies the stubs from a storage media to a virtual address space.

28. The computer-readable medium of Claim 15, wherein loading stubs includes determining the stubs that handle the selected calls.

29. A software system for executing an executable program to determine the results of such execution without the program being executed in the normal manner by an operating system, the software system comprising:

a manager for obtaining an executable and directing calls that are potentially indicative of malware to an simulator;

a loader for making stubs related to said calls that are potentially indicative of malware available to the simulator;

a simulator for executing calls received from said manager, said execution completed using stubs obtained from said loader; and

storage for storing the results of said simulator or executing calls received from said manager.

30. A software system according to Claim 29, wherein said simulator receives and recalls data during the execution of calls received from said manager.

31. A software system according to Claim 29, further comprising a comparator for comparing calls contained in said executable with calls that are indicative of malware.

32. A software system according to Claim 29, wherein said storage is included in a host hardware platform and wherein said loader is included in a host operating system.

33. A software system for simulating an operating system comprising:

(a) an interface operative to accept an executable and identify calls that are potentially indicative of malware;

(b) a set of abbreviated application program interface handlers that mirror a set of fully-implemented application program interface handlers;

(c) an input/output emulator operative to simulate computer devices that accept input or generate output;

(d) a virtual address space for the storage of stubs, the stubs linked to calls made by the executable; and

(e) a memory management unit for mapping locations in memory to a virtual address space.

34. A software system according to Claim 33, wherein said calls are application programming interface ("API") calls and further comprising: a stack data structure operative to store and recall API calls.

35. A software system according to Claim 33, further comprising an output store for storing behavior signatures generated by said stubs.